Blockchain-Based Self-Tallying Voting System with Software Updates in Decentralized IoT

Gang Han, Yannan Li, Yong Yu, Kim-Kwang Raymond Choo, and Nadra Guizani

Abstract

IoT revolutionizes academia as well as industry. An increasing number of interconnected smart devices are gradually changing the urban lifestyle, among which, voting machines are one of the most widely used smart devices. However, the existing voting protocols for IoT are barely satisfactory, in the sense that most of them are centralized and subject to fairness issues. Moreover, the embedded softwares in voting machines are susceptible to internal vulnerabilities and external attacks. To address these issues, in this article, we propose a framework of a blockchain-based self-tallying voting system with software updates in decentralized IoT, which is fully decentralized and fair. In the proposed system, everyone can compute the final election results by collecting the ballots on the blockchain with equal privilege. Voting machines achieve self-tallying voting functionality in decentralized IoT systems and each entity in the system can obtain the voting results. Vendors deploy smart contracts to publish new patches securely and reliably. We implement a prototype of the proposed framework on laptops and mobile phones respectively to demonstrate its practicality.

Introduction

The Internet of Things (IoT) ecosystem [1, 2] is a network consisting of billions of smart devices such as sensors and actuators, connecting without any human interaction. The embedded software and hardware in the smart devices and the connectivity among them allow the IoT system to collect, share and exchange data. Typical IoT applications such as smart city and smart home make IoT pervasive and closer to people's daily life, and currently, believe it or not, IoT is experiencing a new wave of prosperity and is also changing the urban lifestyle. According to iot-analytics.com, which is a company engaging in intelligence gathering about the IoT industry and publishing valuable digital reports, in August 2018 the number of active IoT devices worldwide has exceeded 7 billion, driven by both consumers and enterprises, and this number is estimated to become 10 billion by 2020 and 22 billion by 2025 (https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/).

Voting machines are one of the most widely used smart devices in IoT. According to Pamela Smith, president of election integrity organization verified voting, voting machines are widely leveraged in e-voting, such as in Louisiana, Georgia, New Jersey and so on (https://www.esecurityplanet.com/network-security/vulnerable-voting-machines-yet-another-iot-device-to-secure.html). The advantages of electronic voting are prominent in the sense that they are convenient and save much energy and cost in casting and tallying the votes. Moreover, thanks to the development of science and technology, the financial costs to produce voting machines are much lower and thus they are widely used in IoT applications, such as the leader election in wireless sensor networks. However, the existing voting protocols suffer from the following defects. First, most of the existing voting schemes are centralized. To be more specific, there is a central party that organizes the whole election and also collects and tallies the ballots. This is naturally not suitable in decentralized IoT applications. A decentralized IoT system overcomes the drawbacks of a centralized IoT framework such as single-point-of-failure, and can also enjoy efficient peer-to-peer communication. Thus, a decentralized voting protocol for decentralized IoT scenarios is more suitable than the centralized protocols. Second, the existing voting systems are subject to fairness issues. In traditional centralized e-voting systems, the central party, which tallied the votes, has the priority to obtain the final result, which is undesirable. Even in self-tallying voting systems, where there is no central party involved and each party can do the tally when all the ballots are cast, it is still non-trivial to deal with fairness issues. To be more specific, the final result in self-tallying voting schemes is computed with all the ballots, thus, the last voter will have the priority to access the voting result ahead of time, which is dubbed adaptive issues. If the last voter refuses to cast their vote, then it is hard for the other users to get the final result, which is the abortive issues. Third, voting machines are vulnerable to malware and can be easily hacked, such as SQL injection flaws and and so on. It is said that voting machines become weak instead of smart when they are connected into the network as some unknown attacks will be carried out by adversaries. It is reported that nearly half of the voting machines suffered from vulnerabilities (https://www.wsj.com/articles/widely-used-election-systems-are-vulnerable-to-attack-report-.finds-1538020802). Therefore, the embedded

Digital Object Identifier: 10.1109/MNET.001.1900439

Gang Han is with the Xi'an University of Posts and Telecommunications; Yannan Li is with the University of Wollongong; Yong Yu is with Shaanxi Normal University; Kim-Kwang Raymond Choo is with the University of Texas at San Antonio; Nadra Guizani is with Purdue University.

0890-8044/20/$25.00 © 2020 IEEE

softwares in voting machines need to be updated to repair the internal vulnerabilities and to resist more external attacks.

Contribution

The aim of this article is to address the issues mentioned above. Specifically, the contributions of this article are three-fold:

- We introduce self-tallying voting protocols in decentralized IoT for the first time. We also handle the software update problem in IoT devices and propose the concept of a self-tallying voting system with software updates in decentralized IoT.
- We take advantage of consortium blockchain [3] to integrate the IoT devices and the vendors (clouds) [4], and propose a framework of a blockchain-based self-tallying voting system with software updates in decentralized IoT, which is fully decentralized and can also achieve fairness. Smart contracts are employed by the vendors to provide secure and reliable patches for the voting machines.
- We implement the proposed framework with a demo on laptops and mobile phones respectively to show the efficiency of our protocol.

Related Work

Traditional Centralized Methods for E-Voting

Recall the traditional paper-ballot voting in real-world elections. A voter asks for a voting card with the candidates' names from an official office, chooses the candidates they support by crossing the corresponding options on the voting card. The voter then puts the voting card in an opaque envelope made of carbon paper and seals it to make a ballot. To cast the ballot, the voter presents the envelope with their identity to the official office, where the voter is checked if registered and cast a vote already. But the office gets nothing about the vote itself. If the voter is registered at the office but does not cast a ballot yet, the office then stamps the envelope. In this way, the imprint of the stamp is visible on both the envelope and the voting card within. The voter posts the stamped envelop anonymously, free of charge, to a counting center who shuffles all received envelops, opens the envelops and counts the voting cards with an imprint of the official stamp.

E-voting is the electronic version of paper-based voting systems, which can be classified into three categories. The first one is supervised voting, also known as off-line voting, in which voting machines are usually located at polling stations but not connected, and voters are supervised physically by independent electoral authorities. The second one is hybrid voting where voters are supervised physically by election officials but the voting machines are Internet-connected. The last one is remote online voting, where the voters are unsupervised by election officials. Typically, the voters cast their ballots via the Internet using a personal computer or mobile phone.

We review a well-know e-voting protocol due to Fujioka, Okamoto and Ohta with an abbreviation FOO protocol [5]. There were no formal security models when this protocol was invented in 1992. However, it is widely believed that the FOO protocol achieves the privacy property in spite of no security proofs for it. As shown in Fig. 1, three entities including an administrator, a counter and voters, are involved in the FOO voting system. One can use a public board to replace the counter in some applications, since what the counter does is to create a list of ballots and publish it. The authority is responsible for checking if a voter has the right to vote and generates a blind signature of the ballot for a voter. Briefly, the FOO protocol consists of the following steps:

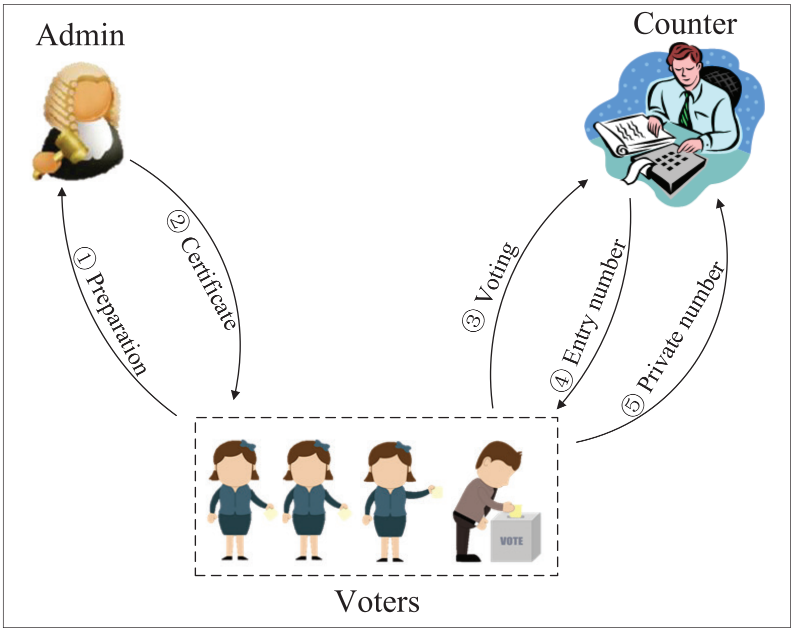

FIGURE 1. FOO protocol.

- A voter prepares their ballot, signs it and keeps a random value private.
- A voter gets a certificate by authenticating themself to the administrator and obtains a blind signature on their ballot.
- A voter submits their ballot along with the administrator's signature to the counter anonymously.
- The counter returns a voting number to the voter.
- Upon closing the election, the voter sends their private random value to the counter anonymously.

Security Requirements of Traditional E-Voting

Some high level and important security properties of e-voting protocols are as follows:

- Eligibility. Only authorized voters can vote, and can only vote once.
- Ballot secrecy. Nobody can figure out how a voter voted, even if the voter tries to prove it.
- Receipt-freeness. After the election, a voter is unable to prove how they voted.
- Coercion-resistance. No one should be able to force a voter to vote in a certain way or abstain from voting.
- Individual verifiability. A voter is able to verify if their ballot has been counted.
- Universal verifiability. Anyone can verify if a tally is correctly computed from the ballots that were cast by legitimate voters.

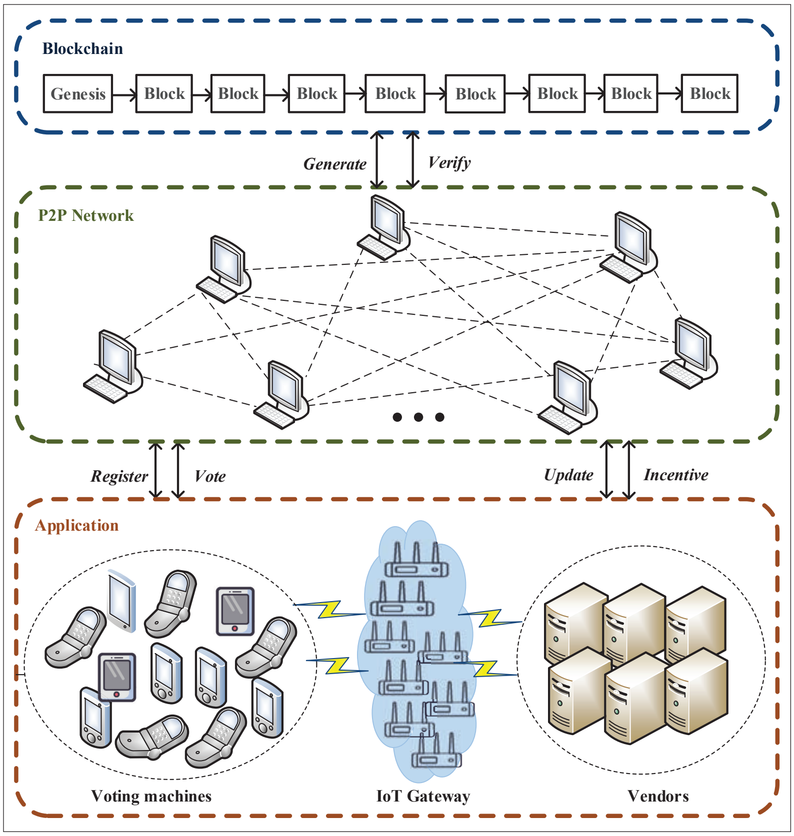

FIGURE 2. System model of blockchain-based self-tallying voting system with software update in decentralized IoT.

- Fairness. No partial results are known ahead of schedule before the election is closed.
- Robustness. Nobody can disrupt an election once it is started.

Cryptographic Building Blocks

A number of novel public-key cryptographic techniques were employed to achieve the target security properties of e-voting protocols mentioned above. We review some widely used techniques as follows.

Commitments: A commitment is a digital equivalent of putting a voting card in an opaque envelope and sealing it. In such a way, nobody else can read the voting card until the envelope is opened (hiding) and the voter cannot modify the content of the voting card (binding) since the voter has committed the vote.

Blind Signatures: A blind signature is useful in applications where a signer signs a message without getting to know the content of the message, and e-voting is such a scenario. In e-voting protocols, a voter commits their voting card. To make sure that only eligible voters cast ballots, the voter needs to authenticate themself to an authority, who signs the committed ballot without knowing the content of the ballot. Here, the authority has to use a blind signature instead of an ordinary digital signature in the sense that the voter does not want to reveal the content of the ballot.

Zero-Knowledge Proofs: Trust is one of the serious issues in e-voting. For example, a voter might encrypt a special message to get an unfair advantage for cheating, or the authority may announce a wrong voting result and so on. Zero-knowledge plays an important role to ensure one has done certain operations correctly without leaking more than the fact, which works in the following way. Take the voter as an example. A ballot, as well as a proof that the ballot was correctly generated, are needed when a voter submits a ballot. The property of zero-knowledge indicates that the proof of the ballot being correctly generated does not reveal the content of the ballot.

Homomorphic Encryption: Homomorphic encryption enables one to perform an additional computation on ciphertexts in such a way that taking two ciphertexts as input and generates a new ciphertext for the "sum" of the two corresponding plaintexts. Homomorphic encryption is quite useful to achieve ballot privacy without affecting counting votes in e-voting protocols.

Cryptographic Counter: A cryptographic counter allows a group of users to increment some specific values privately and robustly. Implementations of these counters rely on (fully) homomorphic encryption and with the help of other primitives such as zero-knowledge proofs. A typical process in voting with the cryptographic counter is as follows. The authority initializes the cryptographic counter and passes it to one of the voters. Each voter increments or randomizes it to vote and shows a proof to indicate their operation. After that, the voter passes the counter to the next one. After all the voters cast their ballots, the authority decrypts and gets the final results.

Mix-Nets: Mix-nets are another important tool to anonymize ballots in an e-voting protocol. Compared with zero-knowledge proofs, mix-nets can deal with ballots with arbitrary formats. A typical implementation of mix-nets in e-voting is to shuffle all the ballots and re-encrypt all the ballots. In such a way, the ballots stay the same even if their ciphertexts are shuffled and the link between the ballots and the voters is hidden.

Bulletin Boards and Blockchain: A public bulletin board is an information repository that is published for anyone in the system to check the validity of the shared data. Bulletin boards are widely used in cryptographic systems for secure verification. In an e-voting system, the bulletin board is responsible for recording some election-relative events, such as the public keys, the encrypted ballots, the tally results and so on. Blockchain is a global bulletin board, which is fully decentralized and tamper-resistant. Anyone in the system can access the blockchain freely to publicly audit the logged data. Such a tool can be a perfect overlay in designing self-tallying voting protocols.

Related Works of Self-Tallying Voting Protocols

The existing self-tallying voting protocols are not quite many. The concept of self-tallying election was proposed by Kiayias and Yung [6]. They proposed two schemes, in which one is suitable for boardroom elections and the other can be adapted to large-scale elections. They also define the security requirements that a self-tallying election needs to satisfy. Later in 2004, Groth [7] proposed a self-tallying voting scheme and proved its security in a simulation-based model. Unfortunately, it does not offer efficient fault correction in the scheme. Hao *et al.* [8] presented a self-tallying voting scheme based on an anonymous veto

protocol, which is obtained from a clever mathematical transformation. Khader *et al*. [9] focused on the fairness issues in [8] and added another recovery phase, in which they ignored the ballots from the abortive voters. In 2017, McCorry *et al*. [10] proposed a blockchain-based self-tallying boardroom voting model, in which smart contract (http://ethereum.org/ethereum.html) is leveraged. They developed a prototype for the voting system.

Related Works of Blockchain-Based E-Voting Protocols

The contribution of blockchain in an e-voting system can be categorized into two aspects, to be a public bulletin board and to replace the central party. Most of the existing works on blockchain-based e-voting systems are in the first scope [11, 12]. In such scenarios, the function of the blockchain is elaborated above. Few works are in the second scope [10, 13] and the basic ideas are analyzed in the previous subsections.

Blockchain-Based Self-Tallying Voting System with Software Update in Decentralized IoT

In this section, we first present the system model of a blockchain-based self-tallying voting system with software updates in decentralized IoT. Then we introduce the workflow of the whole system.

System Model

As illustrated in Fig. 2, there are five entities involved in the system model, which are a blockchain [14], a peer-to-peer (P2P) network, voting machines, IoT gateways and vendors. The details are listed below.

Blockchain: In our system, we leverage a consortium blockchain between the vendors and IoT devices, in which the transaction throughput is high and the block generation time is adjustable [2]. Different from its public counterpart, partial control works on a consortium blockchain. In consortium blockchains, the nodes need registration to participate in the system and thus it is also applicable to regulated purposes. Also, the required computational power is lower than that in the public blockchain. Blockchain in our system serves as a public bulletin board, which can be accessed by anyone in this system, to record all the modifications in the lifetime. First, the blockchain needs to enroll the voting machines when they first register in the system and realize device management. We define such transactions as $T_{register}$. Second, the blockchain needs to collect the ballots from voting machines and make everyone in the system to get the final result at the same time to achieve self-tallying and fairness property. Such transactions related to Vote phase are denoted as T_{vote}. Third, in the software update phase, the vendors publish the patches on the Blockchain and issue incentives to some designated miners. These transactions are written as T_{update}.

P2P Networks: A P2P network consists of lots of authorized terminals, which are called the miners. The miners are supposed to possess plenty of computational capacity and storage resources, whose responsibility is to generate new blocks with their "efforts." All the miners run a managed consensus algorithm, such as Practical Byzantine Fault Tolerance (PBFT) or Tendermint [2], to achieve consistency and stability in the system. In our system, the P2P network is accessible by any user of the system and generates new blocks with different types of transactions.

Voting Machines: Voting machines are the IoT devices that can cast ballots with the embedded software provided by the vendors. They need to register when they first join the system. The gateway will generate an IP address to identify each voting machine and send a $T_{register}$ transaction to the blockchain. The voting devices can generate a vote and cast ballots when needed. Also, the voting machines need to update the embedded software with the patches in the blockchain provided by the vendors.

IoT Gateway: IoT gateway is the key component to integrate the IoT devices. IoT gateway is composed of many routers, which perform multiple important functions, such as interconnect the devices, collect and forward captured data, translate the sensor protocol and more. IoT gateway is also the bridge between the voting machines and vendors in our system.

Vendors: Vendors are the service providers of the voting machines, which also provide continuous revamp of their products and service offerings. When the vendors publish new updated software or patches for the voting machines, they generate a smart contract into the blockchain, and the corresponding voting machines will update the software embedded in the blockchain.

Security Requirements of Blockchain-Based Self-Tallying Voting with Software Updates

A blockchain-based self-tallying voting system with software updates in decentralized IoT needs to satisfy the following requirements.

- Decentralization: There is no central party in the system. All the nodes in this system are equal in the sense that each node enjoys the same rights and services.
- Fairness: No partial-tallying results are revealed before the end of the whole voting process [6].
- Self-Tallying: After all the voting machines cast their ballots, any entity in the system can do the tallying with the collection of all the ballots.
- Ballot Secrecy: The knowledge of the ballots can only be accessible to the coalition of a subset of the other voting machines. The secrecy level is adjustable by choosing the proper size of the subset.
- Software Update: The embedded software in the voting machines can be updated with the secure and reliable patches published by the vendors.

Workflow

The details of a blockchain-based self-tallying voting system with software updates in decentralized IoT are as follows. As shown in Fig. 3, three phases are involved in the whole workflow, namely Register, Vote and Update.

Register: All the voting machines need to register when they first enroll in the system. W.l.o.g, suppose there are n voting machines, VD_1, ···, VD_n, in our system. Each $VD_i (i \in n)$ registers to

IoT gateway to apply for a unique IP address in the intranet. Then with this IP address as a random seed, each voting machine $VD_i(i \in n)$ generates a public/secret key pair $(pk_i, sk_i)(i \in n)$ and registers the $pk_i(i \in n)$ to the blockchain. To be more specific, the voting machine VD_i generates a transaction $T_{register}$ with its pk_i as well as the IP address, and also computes a simple zero-knowledge proof with the secret key sk_i to prove the possession of pk_i, and then puts $T_{register}$ in the blockchain. After all the voting machines have registered their public keys, each VD_i can locally compute the aggregated public key PK with all the legitimate $pk_i(i \in n)$.

Vote: In the *Vote* phase, four sub-phases are included, they are *Commit*, *Cast*, *Tally*, and *Recover*. Inthe *Commit* phase, each voting machine VD_i chooses a vote v_i and generates a corresponding commitment for v_i. Then it puts the commitment into the transaction T_{commit} and logs it into the blockchain. Specifically, the commitment is generated with the selected vote v_i and all the other devices' public key $pk_j(j \neq i)$. In this way, we can ensure the fairness, as if the voting machines are hacked by malware or break down so that they are not able to cast their ballots in the *Cast* phase, we can still recover the vote according to the commitment with the help of all the other voting machines' secret key and get the final voting results. Here, VD_i can also choose some of the voting machines it trusts to generate the commitment, then a subset, rather than all, of the voting machines are required to recover the vote. After the *Commit* phase, the set of the voters cannot be changed, since the corresponding public key are bound in the commitment. In the time slice of *Cast*, each voting machine VD_i randomizes their selected vote v_i with its own secret key sk_i and other voting machines' public key $pk_j(j \neq i)$ to form a secret ballot aiming at protecting the privacy of the vote. Then each VD_i casts the ballot into the blockchain. In this way, after all the voting machines' ballots appear on the blockchain, each entity can collect the ballots in the system and compute the final voting result in the *Tally* phase, as the hiding factor in each secret ballot can be removed when they are aggregated. If some of the registerred voting machines do not cast the ballot, the *Recover* phase can be activated, in which several other voting machines are required. To be more specific, suppose a voting machine VD_i computes its commitment with all the other $pk_j(j \neq i)$, and is injected with malware by some hacker that it cannot cast the ballot in the Vote phase. Then all the $VD_j(j \neq)$ with their $sk_j(j \neq i)$ and pk_i can get together to decommit the commitment and recover a vote to do the tally.

Update: The vendors provide secure and reliable updates for voting machines. When the vendors publish new patches for the voting machines, they will generate a smart contract on the blockchain with their signatures for authentication. Voting machines can download from the smart contract and install them for the update.

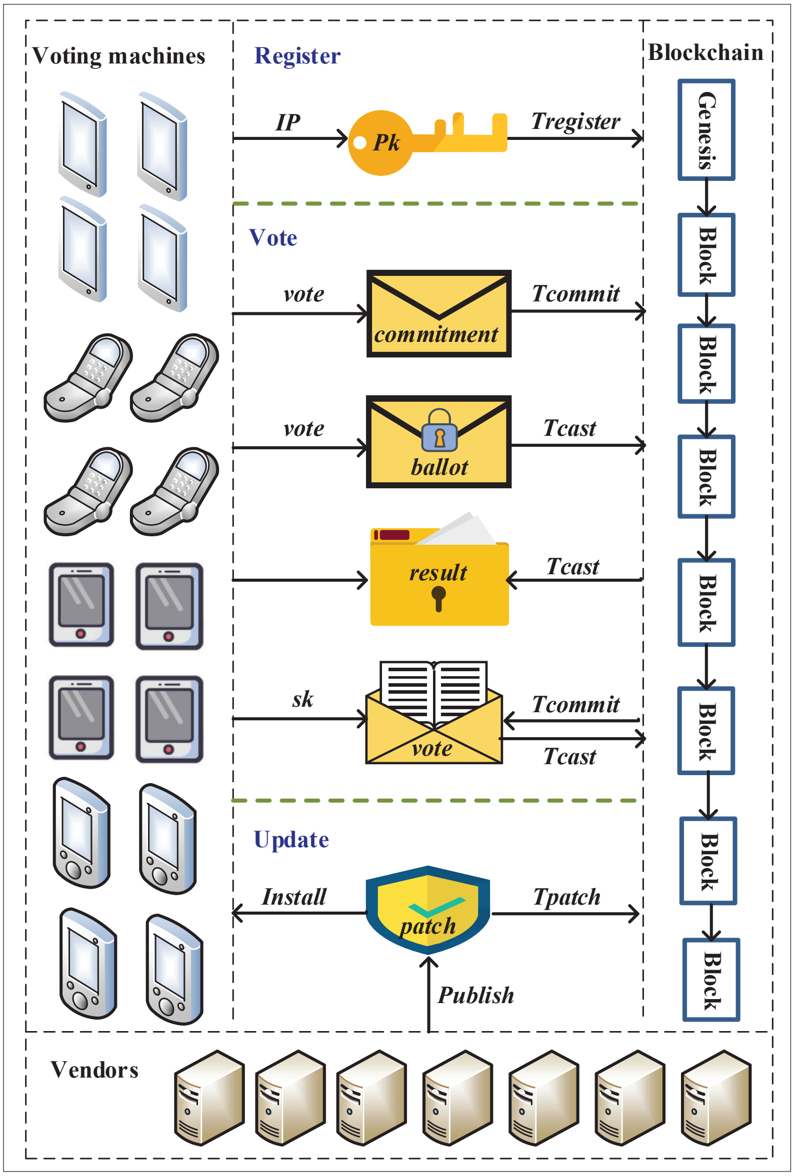

FIGURE 3. Workflow of blockchain-based self-tallying voting system with software update.

Analysis

Compared to the existing solutions, our proposed framework has the following merits.

- Decentralization: The proposed framework is fully decentralized. Based on the blockchain overlay, there is not any central party in the system, which eliminates the risk of a single-point-of-failure.
- Fairness: The protocol is fair with the help of the commitment, homomorphic encryption and zero-knowledge proofs.
- Self-Tallying: The voting protocol is self-tallying based on the algebraic manipulation if all the users follow the process.
- Ballot Secrecy: The ballot secrecy is guaranteed by the encryption of the ballots. The ballot secrecy is achieved if the underlying encryption scheme satisfies indistinguishability.
- Software Update: The protocol can support software updates by the vendors via the blockchain by its design.

Performance and Evaluation

In this section, we elaborate on the simulation results of the proposed construction. In the experiments, we first implement the protocols on a desktop. Then we test the efficiency of the con-

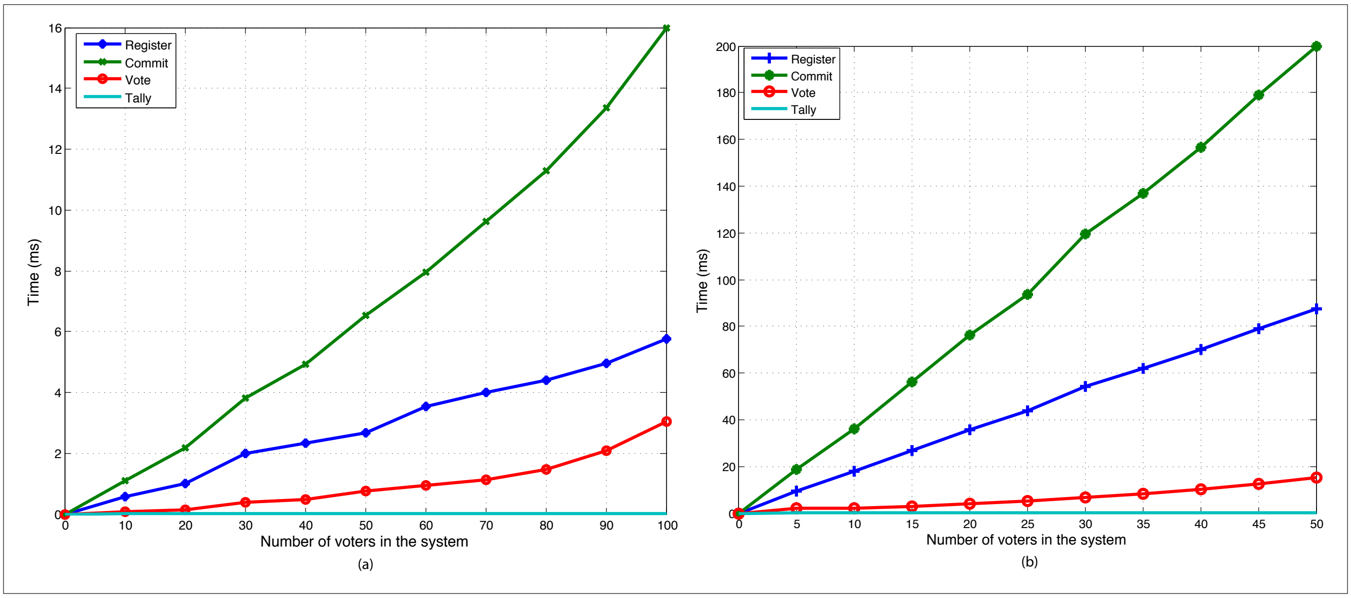

FIGURE 4. Implementation results: a) running time on the laptop; b) running time on the raspberry Pi.

struction on a Raspberry Pi for a better simulation of the IoT voting devices.

Environment

The desktop is equipped with an Intel® Core™ i7-7700 CPU 3.6 GHz and 16 GB RAM, whose operating system is Ubuntu 16.04.6 LTS. The Raspberry Pi 3 Model B+ is loaded with a 64-bit quad-core ARM v7 processor rev 4 and 1 GB LPDDR2 SDRAM. The operating system is Ubuntu Mate 16.04.6 LTS. The project is written in C language with the GMP library (https://gmplib.org/) to deal with the operations among large integers. The modulus is fixed as a 1024-bit prime.

Experiment Results

We test the efficiency of the algorithms including *Register*, *Commit*, *Vote*, and *Tally*. For each algorithm, we repeat 50 times and get the average time consumption. We can see from the Figs. 4a and 4b that the time consumption for each algorithm is linear with the number of voters in the system both on a laptop and a Raspberry Pi, which is consistent with the empirical analysis, as the more voters are involved in the system, the more calculations are needed in each algorithm. The most expensive algorithm is *Commit*, which takes 15.989 ms for 100 voters on a laptop and 199.878 ms for 50 voters on a raspberry Pi, respectively. The cheapest algorithm is *Tally*, as after the ballots are collected, only several multiplications are needed to tally the results; it costs less than 0.002 ms for 100 voters on a laptop and 0.02 ms for 50 voters on a raspberry Pi to tally the results of the voting. To cast ballots, it takes less than 20 ms for 50 voters serially on a raspberry. For the voters on a laptop to cast the ballots, it costs less than 3 ms for 100 voters, which is very efficient. To register as valid voters, the time consumption is almost 90 ms on a raspberry Pi for 50 voters, and this is only a one-time phase. When it comes to the laptop, the corresponding time consumption for 100 voters is less than 6 ms, which enjoys high efficiency.

Conclusion

The Internet of things has been leading digital innovation and industrial revolution in recent years. Voting machines, one of the most widely used smart devices, have attracted our attention. In this article, we focused on the issues in voting protocols in decentralized IoT and proposed a framework of blockchain-based self-tallying voting systems with software updates in decentralized IoT. We developed a prototype to test the proposed framework, which proves it is practical and efficient.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (61872229, 61960206014) and the Basic Research Program of Qinghai Province (2020-ZJ-701). Yannan Li and Yong Yu are the corresponding authors.

References

[1] X. Du and H. H. Chen, "Security in Wireless Sensors Networks," *IEEE Commun. Mag.*, vol. 15, no. 4, Aug. 2008, pp. 60–66.

[2] Y. Yu *et al.*, "Blockchain-Based Solutions to Security and Privacy Issues in the Internet of Things," *IEEE Wireless Commun.*, vol. 25, no. 6, 2018, pp. 12–18.

[3] Y. Li *et al.*, "Traceable Monero: Anonymous Cryptocurrency with Enhanced Accountability," *IEEE Trans. Dependable and Secure Computing*, doi: 10.1109/TDSC.2019.2910058, 2019.

[4] Y. Yu *et al.*, "Attribute-Based Cloud Data Integrity Auditing for Secure Outsourced Storage," *IEEE Trans. Emerging Topics in Computing*, doi:10.1109/TETC.2017.2759329, 2017.

[5] A. Fujioka, T. Okamoto, and K. Ohta, "A Practical Secret Voting Scheme for Large Scale Elections," *Proc. Auscrypt 1992*, LNCS 718, 1992, pp. 244–51.

[6] A. Kiayias and M. Yung, "Self-Tallying Elections and Perfect Ballot Secrecy," *Proc. Int'l. Workshop on Public Key Cryptography*, Springer, Berlin, Heidelberg, 2002, pp. 141–58.

[7] J. Groth, "Efficient Maximal Privacy in Boardroom Voting and Anonymous Broadcast," *Proc. Int'l. Conf. Financial Cryptography*, Springer, Berlin, Heidelberg, 2004, pp. 90–104.

[8] F. Hao, P.Y.A. Ryan, and P. Zieliski, "Anonymous Voting by Two-Round Public Discussion," *IET Information Security*, vol. 4, no. 2, 2010, pp. 62–67.

[9] D. Khader *et al.*, "A Fair and Robust Voting System by Broadcast," Lecture Notes in Informatics (LNI), *Proc. Series of the Gesellschaft fur Informatik (GI)*, 2012, pp. 285–99.

[10] P. McCorry, S.F. Shahandashti, and F. Hao, "A Smart Contract for Boardroom Voting with Maximum Voter Privacy," *Proc. Int'l. Conf. Financial Cryptography and Data Security*, Springer, Cham, 2017, pp. 357–75.
[11] B. Yu *et al*., "Platform-Independent Secure Blockchain-Based Voting System," *Proc. Int'l. Conf. Information Security*, Cham, 2018, Springer, pp. 369–86.
[12] N. Kshetri and J. Voas, "Blockchain-Enabled E-Voting," *IEEE Software*, vol. 35, no. 4, 2018, pp. 95–99.
[13] X. Yang *et al*., "Decentralized Voting: A Self-Tallying Voting System Using a Smart Contract on the Ethereum Blockchain," *Proc. Int'l. Conf. Web Information Systems Engineering*, Springer, Cham, 2018, pp. 18–35.
[14] N. Satoshi, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.

Additional Reading

[1] J. Katz, "Digital Signatures," *Springer Science and Business Media*, 2010.

Biographies

Gang Han is a lecturer at the School of Cyberspace Security, Xi'an University of Posts and Telecommunications, China. He received his Ph.D. degree in electronic science and technology from Northwestern Polytechnical University in 2020. His research interest is information security.

Yannan Li is currently a Ph.D. candidate at the University of Wollongong, Australia. Her research interests are blockchain and cloud security.

Yong Yu is a Professor at Shaanxi Normal University, China. He received his Ph.D. degree in cryptography from Xidian University in 2008. He has authored over 100 referred journal and conference papers. His research interests are blockchain and cloud security.

Kim-Kwang Raymond Choo holds a Cloud Technology Endowed Professorship at the University of Texas at San Antonio, USA. His research interest is information security.

Nadra Guizani is a lecturer at Gonzaga University and a Ph.D. computer engineering student at Purdue University, completing a thesis on prediction and access control of disease spread data on dynamic network topologies. Her research interests include machine learning, large data analysis, and prediction techniques.